# Anomaly Detection

Dataset $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$


## Density estimation.

$p(x_{test}) < \varepsilon \rightarrow$ flag anomaly.

$\geq \varepsilon \rightarrow$ ok.

how anamoly ?

Ex. Fraud detection.

- $x^{(i)}$ = features of user i's activities
- model $p(x)$ from data.
- identify unusual users by checking $p(x) < \varepsilon$.

Ex. Manufacturing

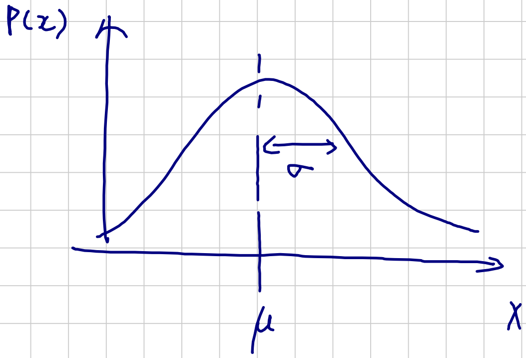
Ex. Monitoring computers in a data centre.

- $x^{(i)}$ = features of machine i.
- $x_1$ = memory use, $x_2$ = number of disk accesses / sec.
- $x_3$ = CPU load, $x_4$ = CPU load / network traffic.

## Gaussian Distribution

$x \in \mathbb{R}$ , $x \sim \mathcal{N}(\mu, \sigma^2)$ "normal"

mean (for $\mu$), variance. (for $\sigma^2$)

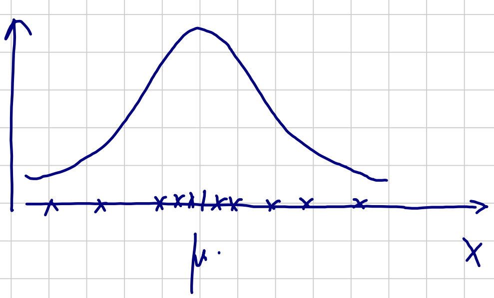



$$p(x; \mu, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

## Parameter Estimation

Dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ , $x^{(i)} \in \mathbb{R}$.

If we expect. $x^{(i)} \sim N(\mu, \sigma^2)$




$$\hat{\mu} = \frac{1}{m} \cdot \sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} [x^{(i)} - \mu]^2$$

$m-1$. (sometimes)

## Algorithm

Training set. $\{x^{(1)}, \ldots, x^{(m)}\}$ . Each example is $x \in \mathbb{R}^n$.

$$p(x) = \prod_{j=1}^{n} p(x_j ; \mu_j, \sigma_j^2)$$

$x_j \sim N(\mu_j, \sigma_j^2)$

1. Choose features $x_i$ that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1 \cdots \mu_n$. $\sigma_1^2, \cdots \sigma_n^2$ $(j = 1 \cdots n)$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \quad ; \quad \sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

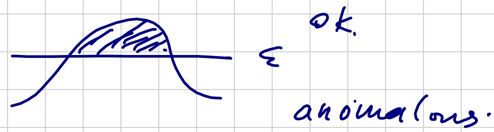
$$\underline{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

3. Given a new example $x$

$$p(x) = \prod_{j=1}^{n} p(x_j ; \mu_j , \sigma_j^2 )$$

$$= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left( - \frac{(x_j - \mu_j)^2}{2\sigma_j^2} \right) \overset{?}{<} \varepsilon \rightarrow \text{anamoly}$$

"Assuming a Gaussian distribution, determine anamoly based on a threshold $\varepsilon$"




## Developing and Evaluating an Anomaly Detection

· real number evaluation.

### Algorithm evaluation

. fit model $p(x)$ on training set. $\{x^{(1)}, \ldots x^{(m)}\}$

. on a CV / test $x$, predict. $y = \begin{cases} 1, & p(x) < \varepsilon \quad \text{(anomaly)} \\ 0, & p(x) > \varepsilon \quad \text{(normal)} \end{cases}$

↳ skewed data set. ( should true/positive, precision/recall, F1 score )

## Anomaly Detection v.s. Supervised Learning

Examples

$1 <$ $\underline{0}$ (negative examples) $\qquad \underline{1}, \quad \underline{0}$

↓

fit $p(x)$

Many diff. types of anomalies

Enough positive examples for algorithm to get a sense of what positive examples are likely.

## Choosing what Features to use

- Non-guassian $\rightarrow$ $\log(x+c)$ , $x^{\frac{1}{a}}$ . ...

  constant

  (transformation)

- error analysis for anomaly detection
  - Want: $p(x)$ large for normal examples $x$

    $p(x)$ small for anomalous examples $x$.

  - common problem: $p(x)$ is comparable (say, both large) for normal and anomalous examples.

## Multivariate Guassian.

$x \in \mathbb{R}^n$. Don't model $p(x_1)$, $p(x_2)$, .., etc. separately.

Model $p(x)$ all in one-go.

Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$. [covariance matrix].

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

$\hookrightarrow$ determinant.

$\Sigma = \begin{pmatrix} 0.6 & 0 \\ 0 & 0.6 \end{pmatrix}$ $\rightarrow$ variance shrinked $\rightarrow$ narrower distribution

$\Sigma = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ flatter.

Anomaly Detection with the multivariate Guassian.

1) Fit model $p(x)$ by setting

$$\mu = \frac{1}{m}, \quad \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu) \cdot (x^{(i)} - \mu)^T.$$

2) Given a new example $x$, compute

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left( -\frac{1}{2} (x-\mu)^T \cdot \Sigma^{-1} (x-\mu) \right).$$

anomaly $\iff$ $p(x) < \varepsilon$.

· Original model $p(x) = \prod p(x_i, \mu_i, \sigma_i^2)$ $\leftrightarrow$ $p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} \cdot (x-\mu)\right)$

special case.

where. $\Sigma = \begin{pmatrix} \sigma & & \emptyset \\ & \ddots & \\ \emptyset & & \sigma \end{pmatrix}$

· When to use?

original model.

$\prod p(x; \mu, \sigma)$

· Manually create features to capture anomalies where $x_1, x_2$ take unusual combinations of values. $x_3 = \frac{x_1}{x_2}$

· $m$ can $< n$.

multivariate Guassian

$$p = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} (x-\mu)\right)$$

· Automatically captures correlations between features.

· Computationally expensive.

· $\Sigma$ invertible $\iff$ $m > n$.

($m \geq 10n$ in practice)

$\hookrightarrow$ redundant features when $n \uparrow$

# Recommender system (use content of movie to predict ratings)

1. Problem formulation: predict users movie rating → generate recommendations

2. Content-based recommendation

   - for each movie, form "feature vector" $= \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \begin{matrix} \to & 1. \\ \to & \text{romance} \\ \to & \text{action} \end{matrix}$

   - for each user $j$, learn parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user $j$ as rating movie $i$ with $(\theta^{(j)})^T x^{(i)}$ starts.

3. Formulation.

   - $r(i,j) = 1$ if user $j$ rated move $i$. ($\emptyset$ otherwise)
   - $y^{(i,j)} =$ rating "

   - $\theta^{(j)}$ = parameter vector for user $j$.
   - $x^{(i)}$ = feature vector for move $i$.

   - For user $j$, movie $i$, predictive $(\theta^j)^T \cdot x^i$.

   - $m^{(j)}$ = # moves rated by user $j$

   To learn $\theta^{(j)}$:

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left[ (\theta^j)^T x^{(i)} - y^{(i,j)} \right]^2$$

$$+ \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} (\theta_k^{(j)})^2, \qquad \theta^j \in \mathbb{R}^{n+1}.$$

   constant, can pull out.

4. Optimization.

   ① $$\underbrace{\min_{\theta^{(1)} \ldots \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} [\quad]^2 + \frac{\lambda}{2} \sum\sum (\theta_k^j)^2}_{J(\theta^{(1)}, \theta^{(2)}, \ldots \theta^{(n)})}$$

   ② gradient descent.

# Feature Learning: Collaborative Filtering

- Given a dataset, but no knowledge of each movie.
- However, has user data $\theta$ parameter.

## 1. Formulation

Given. $\theta^{(1)}, \dots \theta^{(n_u)}$. to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j: r(i,j)=1} \underbrace{\left[ (\theta^{j})^T x^{(i)} - y^{(i,j)} \right]^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2}$$

sum over all movies.

$$\min \frac{1}{2} \sum_{i=1}^{n_m} \left( \qquad \right)$$

## 2. Summary

- Given $x^{(1)}, \dots, x^{(n_m)}$ [and movie ratings] $\xrightarrow{\text{est.}}$ $\theta^{(1)} \dots \theta^{(n_u)}$
- Given $\theta^{(1)}, \dots, \theta^{(n_u)}$ $\rightarrow$ $x^{(1)} \dots x^{(n_m)}$

Can cascade to improve.

## 3. Collaborative Filtering Algorithm

- Minimize over $(i,j): r(i,j) = 1$.
- $x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$ $\rightarrow$ No need to hardcode $x_0 = 1$, since we our generating the features.

1. Initialize $x^{(1)}, \dots, x^{(m)}$; $\theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values.
2. Min. $J(x^{(1)}, \dots x^{(n_m)}, \theta^{(1)}, \dots \theta^{(n_u)})$ using gradient descent.
3. For a user with parameters $\theta$, and a movie w/ learned features $x$, predict a star rating of $\theta^T x$.

# Vectorization: Low Rank Matrix Factorization

$$Y = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \rightsquigarrow y^{(i,j)}$$ move $i$, user $j$

Predicted ratings: $\left[\ \theta^{(j)T} \cdot x^{(i)}\ \right]$. , $j = 1 \cdots n_u$, $i = 1 \cdots n_m$.

$$X = \begin{pmatrix} - & x^{(1)T} & - \\ - & x^{(2)T} & - \\ & \vdots & \\ - & x^{(n_m)T} & - \end{pmatrix}$$ $\rightsquigarrow$ each movie example in rows.

$$\Theta = \begin{pmatrix} - & \theta^{(1)T} & - \\ - & \theta^{(2)T} & - \\ & \vdots & \\ - & \theta^{(n_u)T} & - \end{pmatrix}$$

$\boxed{Y = X\,\Theta^T}$ → low rank matrix factorization.

## Finding Related Movies

- For each product $i$, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$
  - might not be human understandable.
- How to find <u>movies $j$</u> related to <u>movie $i$</u>?

small $\|x^{(i)} - x^{(j)}\|$ → two movies are "similar", share similar viewers

# Mean Normalization

New users: $\quad \underbrace{\min}_{\;} \quad \overbrace{\qquad\qquad}^{0} + \frac{\lambda}{2} \sum\sum (x_k^{(i)})^2 + \frac{\lambda}{2} \sum\sum (\theta_k^{(j)})^2$

$\theta = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$

$\theta^T \cdot x = 0 \longrightarrow$ rate every $\emptyset$.

Mean normalization $\rightarrow$ subtract average of movie

$\rightarrow$ normalize each row to $\emptyset$.
(movie)

$\downarrow$ learn $\theta^{(j)}, x^{(i)}$.

For user $j$, movie $i$. predict:

$$\underbrace{(\theta^{j})^T \cdot (x^{(i)})}_{0} + \underline{\underline{\mu_i}} .$$